(19)日本国特許庁（ＪＰ）　(12) **公開特許公報**（Ａ）　(11)特許出願公開番号

特開平6－223096

(43)公開日　平成６年(1994)８月12日

(51)Int.Cl.[5]	識別記号	庁内整理番号	Ｆ Ｉ	技術表示箇所
Ｇ０６Ｆ　15/31		Ｍ　7343－5Ｌ		
11/10	３３０　Ｑ	7313－5Ｂ		
// Ｈ０３Ｍ　13/00		8730－5Ｊ		

審査請求　未請求　請求項の数７　ＯＬ　（全 12 頁）

(21)出願番号　特願平5－9334

(22)出願日　平成５年(1993)１月22日

(71)出願人　000001007
キヤノン株式会社
東京都大田区下丸子３丁目30番２号

(72)発明者　岩村　恵市
東京都大田区下丸子３丁目30番２号キヤノン株式会社内

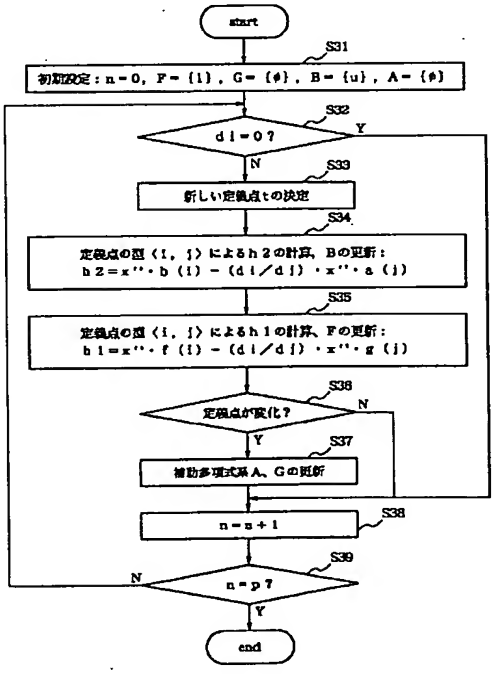
(74)代理人　弁理士　丸島　儀一

(54)【発明の名称】　多項式系導出装置及びその方法

(57)【要約】

【目的】　代数幾何符号の復号のための多次元配列を生成する最小多項式の導出を高速に実行する。

【構成】　与えられた多次元配列を生成する最小多項式系Ｆを求めるために、多項式系Ｆを更新する際に、$d_f^{(i)}$ を直接計算せず、新たに導入した多項式系Ｂに属する多項式の最高次係数 d_i を用いて、ステップＳ３４において多項式系Ｂを更新し、ステップＳ３５において多項式系Ｆを更新する。



1

【特許請求の範囲】

【請求項１】　与えられた多次元配列を生成する最小多項式系を求める多項式系導出装置において、

求める第１の多項式系を記憶するための第１多項式記憶手段と、

当該第１の多項式系と異なる第２の多項式系を記憶するための第２多項式記憶手段と、

前記第１及び第２の多項式記憶手段に初期値を設定する設定手段と、

前記第２多項式記憶手段に記憶された第２の多項式系の所定の次数の係数から得られる値を用いて、前記第１多項式記憶手段に記憶された第１の多項式系を更新する第１更新手段と、

前記第２多項式記憶手段に記憶された第２の多項式系を更新する第２更新手段とを有することを特徴とする多項式系導出装置。

【請求項２】　前記第１更新手段による更新動作と前記第２更新手段による更新動作とを並列に実行することを特徴とする請求項１に記載の多項式系導出装置。

【請求項３】　前記第１更新手段と前記第２更新手段とを共通の回路手段によって実現すること特徴とする請求項１に記載の多項式系導出装置。

【請求項４】　前記第２更新手段が、$b_n^{(i)}$、$b_m^{(j)}$を前記第２多項式系に属する多項式とし、d_i、d_jを、それぞれ当該多項式$b_n^{(i)}$，$b_m^{(j)}$の最高または最低次の係数として、演算

$$b_q^{(k)} = z^{rs} \cdot b_n^{(i)} - (d_i / d_j) \cdot z^{rt} \cdot b_m^{(j)}$$

を行うことを特徴とする請求項１に記載の多項式系導出装置。

【請求項５】　前記第２更新手段が、$b_n^{(i)}$、$b_m^{(j)}$を前記第２多項式系に属する多項式とし、d_i、d_jを、それぞれ当該多項式$b_n^{(i)}$，$b_m^{(j)}$の最高または最低次の係数として、演算

$$b_q^{(k)} = d_j \cdot z^{rs} \cdot b_n^{(i)} - d_i \cdot z^{rt} \cdot b_m^{(j)}$$

を行うことを特徴とする請求項１に記載の多項式系導出装置。

【請求項６】　与えられた多次元配列を生成する最小多項式系を求める多項式系導出方法において、

求める第１の多項式系を記憶するための第１メモリと、

当該第１の多項式系と異なる第２の多項式系を記憶するための第２メモリとに初期値を設定し、

前記第２メモリに記憶された第２の多項式系の所定の次数の係数から得られる値を用いて、前記第１メモリに記憶された第１の多項式系を更新し、

前記第２メモリに記憶された第２の多項式系を更新することを特徴とする多項式系導出方法。

【請求項７】　前記第１の多項式系の更新動作と前記第２の多項式系の更新動作とを並列に実行することを特徴とする請求項６に記載の多項式系導出方法。

【発明の詳細な説明】

2

【０００１】

【産業上の利用分野】本発明は、ディジタル通信系及びディジタル記憶系において通信路または記憶媒体で受けた誤りを、誤り訂正符号を用いて受信側で自動的に訂正する誤り訂正に関し、特に誤り訂正符号として代数幾何符号を用いる場合の復号において、与えられた多次元配列を生成する最小多項式系を求める多項式系導出装置及びその方法に関する。

【０００２】

【従来の技術】従来、ディジタル通信系及びディジタル記憶系において通信路または記憶媒体で受けた誤りを、受信側で自動的に訂正する誤り訂正符号として、リード・ソロモン（ＲＳ）符号やＢＣＨ符号がよく知られ、コンパクト・ディスクや衛星通信等において実際に装置化され使用されている（参考文献[1] 参照）。

【０００３】しかし、最近は、代数曲線論を駆使した代数幾何符号（参考文献[1]-[4] 参照）と呼ばれる符号の研究が盛んに行われている。代数幾何符号は、前述のＲＳ符号やＢＣＨ符号をその１クラスとして含む非常に適用範囲の広い符号系であり、従来の符号よりもよい性質を持つ新符号が存在することが徐々に明らかになってきている（参考文献[5][6]参照）。

【０００４】このように優れた符号が発見される中で、恒に問題となっていたのは復号法の問題であり、その効率的な復号アルゴリズムの開発が急がれたが、１９８９年にJustesenらのグループが一般化Peterson algorithm（参考文献[7] 参照）を提案するまで、長い間有効な復号アルゴリズムは全く得られなかった。また、Justesenらは、与えられた有限の大きさを持つ２次元配列を生成する、記憶素子数最小の２次元線形帰還シフトレジスタを求める効率的なアルゴリズムとして、阪田によって提案された阪田アルゴリズム（参考文献[8] 参照）を、彼らの一般化 Peterson algorithm に応用することで、少ない計算量で高速に誤り位置関数を導出できることを示した。

【０００５】しかし、一般化Peterson algorithmは、その訂正能力に制約があったために、Skorobogatovらはより高い訂正能力を保証する Modified decoding algorithm（参考文献[9] 参照）を提案した。また、阪田アルゴリズムには Modified decoding algorithmに対しては直接関係のない無駄な演算が多く存在したために、神谷らは、阪田アルゴリズムを Modified decoding algorithm に沿った形（参考文献[11][12]参照）に修正したアルゴリズムを１９９２年に提案した。

【０００６】一方、従来用いられているＲＳ符号やＢＣＨ符号の復号法として、Peterson法と、バーレカンプ・マッセイ（ＢＭ）法及びユークリッド（Ｅｕ）法がよく知られている。前述の一般化 Peterson algorithm はＲＳ符号の復号に用いられるPeterson法の拡張になっている。また阪田アルゴリズムは２次元ＢＭ法とも呼ばれ、

3

ＲＳ符号の復号に用いられるＢＭ法（以後、１次元ＢＭ法と呼ぶ）の拡張になっていることが知られている。

【０００７】

【発明が解決しようとしている課題】しかしながら、ＲＳ符号の復号に用いられるＥｕ法（以後、１次元Ｅｕ法と呼ぶ）の拡張に相当するアルゴリズムは、これまでに考えられていなかった。

【０００８】また、阪田は、２次元ＢＭ法の拡張として多次元ＢＭ法（参考文献[13]参照）も提案したが、それに対応する多次元Ｅｕ法もまた、考案されていなかった。

【０００９】そこで、本発明では、１次元Ｅｕ法の拡張に相当する２次元Ｅｕ法を提案し、更に、この２次元Ｅｕ法を拡張して、多次元ＢＭ法に対応する多次元Ｅｕ法を提案することを目的とする。

【００１０】

【課題を解決するための手段】上記課題を解決するために、本発明では、与えられた多次元配列を生成する最小多項式系を求める多項式系導出装置に、求める第１の多項式系を記憶するための第１多項式記憶手段と、当該第１の多項式系と異なる第２の多項式系を記憶するための第２多項式記憶手段と、前記第１及び第２の多項式記憶手段に初期値を設定する設定手段と、前記第２多項式記憶手段に記憶された第２の多項式系の所定の次数の係数から得られる値を用いて、前記第１多項式記憶手段に記憶された第１の多項式系を更新する第１更新手段と、前記第２多項式記憶手段に記憶された第２の多項式系を更新する第２更新手段とを具える。

【００１１】

【作用】設定手段により、求める第１の多項式系を記憶するための第１多項式記憶手段及び当該第１の多項式系と異なる第２の多項式系を記憶するための第２多項式記憶手段に初期値を設定し、前記第２多項式記憶手段に記憶された第２の多項式系の所定の次数の係数から得られる値を用いて、第１更新手段によって前記第１多項式記憶手段に記憶された第１の多項式系を更新し、第２更新手段によって前記第２多項式記憶手段に記憶された第２の多項式系を更新する。

【００１２】［参考文献］

[1] 今井：“符号理論”，電子情報通信学会

[2]V.D.Goppa: “Codes on algebraic curves”, Soviet Math.Dokl., 24, pp.170-172, 1981.

[3]V.D.Goppa: “Algebraico-geometric codes”, Math. U.S.S.R. Izvestiya, vol.21, no.1, pp.75-91, 1983.

[4]V.D.Goppa: “Geometry and Codes”, Kluwer Academic Publishers, 1988.

[5]M.A.Tsfasman and S.G.Vladut: “Algebraic-geometric codes”, KluwerAcademic Publishers, 1991.

[6] 三浦：“ある平面曲線上の代数曲線符号”, Preprint.

[7]J.Justesen, K.J.Larsen, E.Jensen, A.Havemose and T.Hoholdt: “Construction and decoding of a class of algebraic geometry codes”, IEEE Trans. Inform. Theory, vol.35, no.4, pp.811-821, July 1989.

[8] 阪田：“与えられた２次元配列を生成する２次元線形帰還シフトレジスタの合成”，信学論（Ａ），vol.J-70A, pp.903-910, 1987.

[9]A.N.Skorobogatov and S.G.Vladut: “On the decoding of algebraicgeometric codes”, IEEE Trans. Inform. Theory, vol.36, no.5, pp.1051-1060, Sep. 1990.

[10]三浦：“代数曲線符号の一般的な復号法(1), (2)”，信学技法，vol.IT89, no.452, IT89-91, 92, pp.61-70, pp.71-80, 1990.

[11]神谷，三浦：“ある種の代数曲線符号に関する Modified decodingalgorithmへの Sakata algorithm の応用について”，信学技法，vol.IT91, no.435, IT91-96, pp.47-54, 1992.

[12]神谷，三浦：“ある種の代数曲線符号に関する再帰的な復号アルゴリズム”，信学技法，vol.IT91, no.505, IT91-116, pp.89-96, 1992.

[13]S.Sakata: “Extension of the Berlekamp-Massey algorithm to Ndimensions”, Information and Computation, vol.84, pp.207-239, 1990

【００１３】

【実施例】

［説明の概要］以下では、まず１次元Ｅｕ法の拡張に相当する２次元Ｅｕ法を提案する。

【００１４】そして、この２次元Ｅｕ法が、２次元ＢＭ法である阪田アルゴリズムやその修正アルゴリズムである神谷アルゴリズムと等価な結果を出力することを示す。更に、２次元Ｅｕ法を拡張し、多次元ＢＭ法に対応する多次元Ｅｕ法を提案する。そして、本発明による多次元Ｅｕ法が、装置化や高速化において、従来の多次元ＢＭ法と異なる効果をもつアルゴリズムであることも示し、その違いを明らかにする。

【００１５】従来知られた多次元ＢＭ法のアルゴリズムは、図２に示す構造を持つ。一方、本発明による多次元Ｅｕ法のアルゴリズムは、図３に示す構造を持つ。多次元ＢＭ法と多次元Ｅｕ法の違いは、前者がステップＳ２２において$df_n^{(1)}$を直接計算し、ステップＳ２５においてその値を用いて多項式系Ｆを更新するのに対し、後者では、$df_n^{(1)}$を直接計算せず、新たに導入した多項式系Ｂに属する多項式の最高次係数d_1を用いて、ステップＳ３４において多項式系Ｂを更新し、ステップＳ３５において多項式系Ｆを更新する点である。

【００１６】上記従来の多次元ＢＭ法が、与えられた多次元配列ｕを生成する最小多項式系Ｆを導くことは、前記文献[8]，[13]に証明されている。よって、ここでは本発明による多次元Ｅｕ法が、多次元ＢＭ法と同じ多項

5

式系Ｆを導くことを以下に示す定理１～定理３により証明することによって、図３の多次元Ｅｕ法から得られる多項式系Ｆも、与えられた多次元配列ｕを生成する最小多項式系であることを示す。以下、従来知られた２次元ＢＭ法及び多次元ＢＭ法を例にとり、本発明による多次元Ｅｕ法を具体的に説明する。

【００１７】［実施例１］まず、以下の準備（用語の説明）の後に、阪田アルゴリズムを示す。このアルゴリズムが、与えられた有限２次元配列ｕを生成する、記憶素子数最小の２次元線形帰還シフトレジスタを合成する多項式系Ｆを導出することの証明は、前記文献[8]になさ＊

6

＊れているので省略する。

【００１８】準備１（詳細は文献[8]参考）：

Σ：あらゆる非負整数ｎ１，ｎ２の対ｎ＝（ｎ１，ｎ２）の集合。

【００１９】ｎ：点と呼ばれ、Ｘ－Ｙ平面上の座標（ｎ１，ｎ２）を持つ点と同一視される。

【００２０】$<_\Gamma$：全順序と呼ばれ、Σ上の点ｎの大小関係を定める。ここでは、点ｎ＝（ｎ１，ｎ２）に対し、$<_\Gamma$に関する次の点を以下のように定める。

【００２１】

$$n+1 := (n1-1,\ n2+1) \quad (n1>0\text{のとき})$$
$$:= (n2+1,\ 0) \quad (n1=0\text{のとき})$$

$<_p$：半順序と呼ばれ、次のように定義される。

【００２２】$m1 \leqq n1$，$m2 \leqq n2$のとき、かつそのときに限って$m \leqq_p n$。

【００２３】$m <_p n$はｍ≦ｎかつｍ≠ｎを意味する。

【００２４】$\Sigma_t^p := \{m \in \Sigma \mid t \leqq_p m,\ m <_\Gamma p\}$

ｕ：ｕは大きさｑの有限部分２次元配列であり、Σ_0^qから体Ｋへの写像として定義される。

【００２５】Ｆ：体Ｋ上の２変数多項式を

【００２６】

【外１】

$$f = \sum_{m \in \Gamma f} f_m \cdot z^m$$

【００２７】とする。

【００２８】ただし、$z^m = x^{m1} \cdot y^{m2}$，$\Gamma f = \{m \in \Sigma \mid f_m \neq 0\}$，$s = LP(f) = \max\{m \mid m \in \Gamma f\}$

多項式の組をＦ＝｛$f^{(0)}, \cdots, f^{(l-1)}$｝と表す。

【００２９】$df_n^{(i)}$：Σ上の点ｎから体Ｋへの写像をｕｎとすると、ＬＰ（ｆ(i)）＝ｓ(i)なる多項式ｆ(i)に対し、

【００３０】

※

$$\triangle F = \Sigma / \bigcup_{k=1}^{l} \Sigma s(k),\quad \Sigma s(k) = \{m \in \Sigma \mid s(k) \leqq_p m\}。$$

【００３８】△：上の条件を満たすときの$\triangle_F$。次のようにして求められる。

【００３９】$\triangle = \cup \triangle_{q-s}$。　$q, s \in \Sigma_0^p$

ただし、$df_q \neq 0$かつＬＰ（ｆ）＝ｓとなるｆ∈Ｖ（u^q）が存在するとき

$\triangle^{q-s} = \{m \in \Sigma \mid m \leqq_p q-s\}$、そうでなければ$\triangle^{q-s} = \phi$。

【００４０】ｈ１：型＜ｉ，ｊ＞のｈ１を次のように定義する。

【００４１】$h1 = z^{r-s(i)} \cdot f^{(i)} - (d_i/d_j) \cdot z^{r-n+m-t(j)} \cdot g(j)$

ただし、ｒ＝（ｒ１，ｒ２），$r1 = \max\{s1^{(i)},\ n1 - s1^{(j)} + 1\}$，$r2 = \max\{s2^{(i)},\ n2 - s2^{(j+1)} + 1\}$，$t^{(j)} = LP(g^{(j)})$，$f^{(i)} \in V(u^n)$，$g^{(j)} \in V(u^m)$，$d_i = df_n^{(i)}$，$d_j = dg_m^{(j)}$。

Ｇ：Ｆの補助多項式系と呼ばれ、Ｇ＝｛$g^{(0)}, \cdots, g^{(l-2)}$｝なる多項式の組。

【００４２】補題１：①で定まる型のｊは、ＬＰ（$f^{(j)}$）＝$s^{(j)}$かつ$p - s^{(i)} \leqq_p (s1^{(j)} - 1,\ s2^{(j+1)} - 1)$となるｊである。

【００４３】補題２：②で定まる型のｋは、ＬＰ（$f^{(k)}$）$<_p$ｔとなるｋである。

【００４４】阪田アルゴリズム

１）ｎ＝（０，０），Ｆ＝｛１｝，Ｇ＝φ

２）Ｆの全ての多項式に対して$df_n^{(i)}$を計算する。

【００４５】３）もし$df_n^{(i)} \neq 0$　となるｆ(i)があ

れば、新しい△と定義点tを定める。

4）そのあらゆる定義点tにおいて以下の手続きを実行する。

【0046】①t＝（$s1^{(i)}$，$s2^{(j)}$）→補題1の多項式h1を作る。

【0047】②t＝（n1－$s1^{(i)}$＋1，n2－$s2^{(j+1)}$＋1）→補題2の型＜k，i＞のh1を作る。

【0048】③t＝（n1－$s1^{(i)}$＋1，$s2^{(j)}$），1≦i≦l－1→型＜j，i＞のh1を作る。

【0049】④t＝（$s1^{(i)}$，n2－$s2^{(j)}$＋1），2≦j≦l→型＜i，j－1＞のh1を作る。

【0050】⑤t＝（n1＋1，$s2^{(j)}$）→h1＝$x^{n1-s1(j)+1}\cdot f^{(j)}$

⑥t＝（$s1^{(i)}$，n2＋1）→h1＝$y^{n2-s2(i)+1}\cdot f^{(i)}$

Fから$df_n^{(i)}\neq 0$の全ての多項式を除去し、新しく求めたh1を全てFに挿入する。

【0051】5）△が変化した場合、新しいFの補助多項式系Gの多項式を旧のGと除去したFの多項式の中から選び出す。

【0052】6）n＝n＋1；n＝pならば終了；さもなければ2）へ

以下に、阪田アルゴリズムと同じ出力を導出する、本発明による新しいアルゴリズムを示す。

【0053】アルゴリズム1

1）n＝（0，0），F＝{1}，G＝φ，A＝{x-1}，B＝{u}

2）Bの全ての多項式の最高次係数が0でなければ、新しい△と定義点tを定める。

【0054】3）そのあらゆる定義点tにおいて以下の手続きを実行する。

【0055】①t＝（$s1^{(i)}$，$s2^{(j)}$）→補題1の型をもつ多項式h2を作る。

【0056】②t＝（n1－$s1^{(i)}$＋1，n2－$s2^{(i+1)}$＋1）→補題2の型＜k，i＞のh2を作る。

【0057】③t＝（n1－$s1^{(i)}$＋1，$s2^{(j)}$），1≦i≦l－1→型＜j，i＞のh2を作る。

【0058】④t＝（$s1^{(i)}$，n2－$s2^{(j)}$＋1），2≦j≦l→型＜i，j－1＞のh2を作る。

【0059】⑤t＝（n1＋1，$s2^{(j)}$）→h2＝$x^{-(n1-s1(j)+1)\cdot b(j)}$

⑥t＝（$s1^{(i)}$，n2＋1）→h2＝$y^{n2-s2(i)+1}\cdot b^{(i)}$

Bから最高次係数が0でない全ての多項式を除去し、新しく求めたh2を全てBに挿入する。

【0060】4）3）で決定された型の多項式h1を作る。また、FからBの除去した多項式に対応する多項式を除去し、新しく求めたh1をFに挿入する。

【0061】5）△が変化した場合、新しいFの補助多項式系Gの多項式を旧のGと除去したFの多項式の中から選び出す。また、新しいBの補助多項式系Aの多項式を旧のAと除去したBの多項式の中から選び出す。

【0062】6）n＝n＋1；n＝pならば終了；さもなければ2）へ

ただし、

A，B：A＝{$a^{(0)},\cdots,a^{(l-2)}$}，B＝{$b^{(0)},\cdots,b^{(l-1)}$}となる多項式の組。

【0063】h2：型＜i，j＞のh2を次のように定義する。

【0064】$h2=z^{r-s(i)}\cdot b(i)-(d_l/d_j)\cdot z^{r-n+m-t(j)}\cdot a(j)$

アルゴリズム1と阪田アルゴリズムの大きな違いは、阪田アルゴリズムが$df_n^{(i)}$を直接計算し、それが全て0であるかどうかを検査するのに対し、アルゴリズム1は$df_n^{(i)}$を直接計算せず、Bに属す多項式$b^{(i)}$の最高次係数が0であるかどうかを検査することであり、そのためにアルゴリズム1は阪田アルゴリズムにない多項式の組A，Bを導入している。

【0065】アルゴリズム1によって得られるFと阪田アルゴリズムによって得られるFが同じ多項式系であることは次のようにして証明できる。

【0066】定理1

式（1）のように定義したun（$n\in\Sigma 0p$）を係数とする多項式uとLP（）＝sである多項式fの積多項式をbとするとき、多項式bのz^{v-n+s}次の係数bv-n+sは式（2）のようになる。ただし、vは任意の整数。

【0067】

$$u=\sum_{n\in\Sigma_0^p} u_n\cdot z^{v-n} \qquad (1)$$

$$b_{v-n+s}=\sum_{m\in\Gamma f} f_m\cdot u_{m+n-s} \qquad (2)$$

証明：式（1）から

【0068】

【外4】

$$b=f\cdot u=\sum_m\sum_i f_m\cdot u_i\cdot z^{v-i+m}$$

【0069】である。

【0070】よって、多項式bのz^{v-n+s}の係数bv-n+sはv－i＋m＝v－n＋sよりi＝m＋n－sとなるので、式（2）のようになる。

【0071】（証明終了）この定理からf（i）∈Fに対応するb（i）＝$f^{(i)}$・uのz^{v-n+s}の係数は阪田アルゴリズムの$df_n^{(i)}$となっていることがわかる。

【0072】また、V（u^n）である多項式$f^{(i)}$を$f_n^{(i)}$と表すと次の定理が成り立つ。

【0073】定理2：式（3），（4）によって更新される多項式を$l_q^{(k)}$，$c_q^{(k)}$とする（q＞r n＞r m）。また、多項式$w_q^{(k)}$を式（5）のように定義すると、$w_q^{(k)}$は式（6）によって更新できる。ただし、rs，rtは任意の整数。また、$l_n^{(i)}$，$l_m^{(j)}$及びc_n

9

$^{(i)}$, $c_m^{(j)}$の初期値は任意の多項式。

$$l_q^{(k)} = z^{rs} \cdot l_n^{(i)} - (d_i / d_j) \cdot z^{rt} \cdot l_m^{(j)} \quad (3)$$

$$c_q^{(k)} = z^{rs} \cdot c_n^{(i)} - (d_i / d_j) \cdot z^{rt} \cdot c_m^{(j)} \quad (4)$$

$$w_q^{(k)} = l_q^{(k)} \cdot e - c_q^{(k)} \cdot d \quad (d, e\text{は任意の多項式}) \quad (5)$$

$$w_q^{(k)} = z^{rs} \cdot w_n^{(i)} - (d_i / d_j) \cdot z^{rt} \cdot w_m^{(j)} \quad (6)$$

証明：式（6）は任意のq及びkに対して成り立つので、次が成り立つ。

【0075】$w_n^{(i)} = l_n^{(i)} \cdot e - c_n^{(i)} \cdot d$

$w_m^{(j)} = l_m^{(j)} \cdot e - c_m^{(j)} \cdot d$

従って、次が言える。

【0076】$w_q^{(k)} = l_q^{(k)} \cdot e - c_q^{(k)} \cdot d$

$= (z^{rs} \cdot l_n^{(i)} - (d_i / d_j) \cdot z^{rt} \cdot l_m^{(j)}) \cdot e$

$- (z^{rs} \cdot cn(j) - (d_i / d_j) \cdot z^{rt} \cdot c_m^{(j)}) \cdot d$

$= z^{rs} \cdot w_n^{(i)} - (d_i / d_j) \cdot z^{rt} \cdot w_m^{(j)}$

（証明終了）阪田アルゴリズムにおいて、$V(u^q)$となる$f^{(k)} = f_q^{(k)} \in F$はh1によって更新され、$g^{(j)}$は$V(u^m)$となる$f^{(j)} = f_m^{(j)}$であるので、h1は式（3）の形式で表現できる。

【0077】また、次の定理が成立する。

【0078】定理3：式（5）において、$l_q^{(k)} = f_q^{(k)}$, $e = u$, $d = z^v \cdot x$、及び$c_{-1}^{(j)} = 1$, $c_0^{(i)} = 0$とすると、deg $w_q^{(k)} \leqq v$において$w_q^{(k)} = b_q^{(k)}$である。このとき、$b_q^{(k)}$は$v - q + s^{(k)}$次の多項式である。

【0079】証明：$l_q^{(k)} = f_q^{(k)}$, $e = u$より、式（5）は$w_q^{(k)} = b_q^{(k)} - c_q^{(k)} \cdot d$となる。また、deg $c_{-1}^{(j)} \leqq 0$, deg $c_0^{(i)} \leqq 0$であれば式（4）より$c_q^{(k)}$は正の次数の多項式であり、$d = z^v \cdot x$であるのでdeg $c_q^{(k)} \cdot d > v$である。よって、deg $w_q^{(k)} \leqq v$において$w_q^{(k)} = b_q^{(k)}$である。

【0080】また、定義から$b_q^{(k)} = u \cdot f_q^{(k)}$の最高次数は$v + s^{(k)}$であるが、このときの$f_q^{(k)}$は$f_q^{(k)} \in V(uq)$であるので$df_i^{(k)} = 0$ (i=0, …, q-1) である。定理1から$df_i^{(k)}$は$b_q^{(k)}$の$z^{v-i+s(k)}$の係数であるので、$b_q^{(k)}$の$v - i + s^{(k)}$次(i=0, …, q-1)の係数は0になり、$b_q^{(k)}$の最高次数は$v - q + s^{(k)}$になる。

【0081】（証明終了）従って、$b_q^{(k)} = w_q^{(k)}$は式（6）によって更新できる。即ち、$b_q^{(k)}$はh2によって更新できる。この場合、定理1，定理3からd_i, d_jは各々多項式$b_n^{(i)}$, $b_m^{(k)}$の$z^{v-n+s(i)}$, $z^{v-m+s(j)}$の係数、即ち、最高次係数である。

【0082】以上から、アルゴリズム1によって得られるFと阪田アルゴリズムによって得られるFが同じ多項式系であることが証明された。阪田アルゴリズムによって得られる多項式系Fは与えられた2次元配列uを生成する最小多項式系であることが文献[8]によって証明されているので、アルゴリズム1から得られる多項式系Fも与えられた2次元配列uを生成する最小多項式系であることがいえる。

10

【0074】

【0083】よって、このアルゴリズムは例えば図1のような装置によって実現できる。まず、アルゴリズム1の1）で設定されている初期値及びその更新値を記憶するメモリと、そこに格納されたBの多項式の最高次係数が0であるかどうかを判断し、0であれば6）を実行し、0でなければ新しい定義点を計算し型を判断する2）を実行する制御回路11と、その型に従って3），4）に示されたh2，h1の演算を行う処理回路12によって実現できる。また、制御回路11は定義点が更新されていればメモリ13中のA，Gの多項式を5）に示されるように新しく選択・更新する。

【0084】ただし、制御回路11と処理回路12は、各種制御手順及び前記アルゴリズムに対応するプログラムをCPUに実行させることによりソフトウェア的に実現することもできるので、分離してある必要はない。また、制御・処理において必要な演算は簡単な整数の乗除算と加減算であるので、特殊な回路・処理は必要としない。また、制御における型の判定及び手順の流れは、予めプログラミングしておいたりROMに焼き付けておいて、必要なときに検索（テーブル・ルックアップ）すれば簡単である。

【0085】また、h1とh2は同時に実行することができるので処理回路をh1とh2独立に持つこともできる。また、h1とh2は同様の処理であるので1つの回路によって実行することしてもよい。また、効果の所で後述するように本アルゴリズムの特徴は並列処理のしやすさであるので、処理回路は1つである必要はない。以上から、アルゴリズム1を実行する回路は簡単に実現することができることがわかる。

【0086】［実施例2］次に神谷らに示された Modified decoding algorithm に適したアルゴリズムと等価なアルゴリズムを考える。まず、以下の準備の後に、神谷らによって示されたアルゴリズムを示す。ただし、以下の準備において述べられていない部分は実施例1の場合と同じである。

【0087】準備2（詳細は文献[12]参照）：

$<_\tau$：ここでは全順序は次のように定める。

【0088】$Q(i) = a \cdot i1 + b \cdot i2$　（a，bは互いに素な自然数，$i = (i1, i2)$）

$Q(m) < Q(n)$が満たされるか、$Q(m) = Q(n)$かつm2 ≦ n2 の時に限って、$m \leqq_\tau n$とし、この全順序$<_\tau$に関してmの次の点を$m+1$と表す。

【0089】$\Sigma_t^p(n) := \{m \in \Sigma \mid m2 \leqq n,\ t \leqq_p m,\ m <_\tau p\}$

u　：uは大きさqの有限部分2次元配列であり、Σ_0^q

11

(2･(a-1)) から体Kへの写像として定義される。

【0090】F　：体K上の2変数多項式を

【0091】

【外5】

$$f = \sum_{m \in \Gamma f} f_m \cdot z^m$$

【0092】とする。

【0093】ただし、$z^a = x^{a1} \cdot y^{a2}$, $\Gamma f = \{m \in \Sigma(a-1) \mid f_m \neq 0\}$, $s = LP(f) = \max\{m \mid m \in \Gamma f\}$

多項式の組を$F = \{f^{(0)}, \cdots, f^{(a-1)}\}$と表す。

【0094】V(u)：Q(p)＜qに対するunの集合を$u^p = \{un \mid n \in \Sigma_0{}^p(2\cdot(a-1))\}$とするとき、$u^p$に対して$df_n{}^{(i)} = 0$ $(n \in \Sigma_s{}^p(a-1+s2))$であれば、$f[u^p] = 0$と表す。このとき、$V(u^p) = \{f$は多項式$\mid f[up] = 0\}$とする。

【0095】△　：△は次のようにして求められる。

【0096】$\triangle = \cup \triangle^{q-s}$

$q, s \in \Sigma_0{}^p(2\cdot(a-1))$

ただし、$df_q \neq 0$かつ$LP(f) = s$となる$f \in V(u^q)$が存在するとき$\triangle^{q-s} = \{m \in \Sigma(a-1) \mid m1 \leqq q1 - s1,\ m2 = q2 - s2\}$、そうでなければ$\triangle^{q-s} = \phi$。

【0097】h3　：型＜i，j＞のh3を次のように定義する。

【0098】$h3 = x^{r1-s1(i)} \cdot f^{(i)} - (d_i/d_j) \cdot x^{r1-n1+m1-t1(j)} \cdot g^{(j)}$

G：Fの補助多項式系と呼ばれ、$G = \{g^{(0)}, \cdots, g^{(a-1)}\}$なる多項式の組。

神谷アルゴリズム

1）$n = (0, 0)$, $F = \{1, y, y^2, \cdots, y^{a-1}\}$, $G = \phi$

2）Fの全ての多項式に対して$df_n{}^{(i)}$を計算する。

【0099】3）もし$df_n{}^{(i)} \neq 0$となる$f^{(i)}$があれば、新しい△と定義点tを定める。

4）そのあらゆる定義点tにおいて以下の手続きを実行する。

【0100】$t = (s1^{(i)}, s2^{(i)}) \rightarrow$型＜i，n2 － i＞のh3を作る。

【0101】$t = (n1 - s1^{(i)} + 1, n2 - s2^{(i)}) \rightarrow$型＜n2 － i，i＞のh3を作る。

【0102】$t = (n1 + 1, n2 - s2^{(i)}) \rightarrow h3 = x^{n1-s1(n2-i)+1} \cdot f^{(n2-i)}$

Fから$df_n{}^{(i)} \neq 0$の全ての多項式を除去し、新しく求めたh3を全てFに挿入する。

【0103】5）△が変化した場合、新しいFの補助多項式系Gの多項式を旧のGと除去したFの多項式の中から選び出す。

【0104】6）n＝n＋1；n＝pならば終了；さもなければ2）へ

12

神谷アルゴリズムと等価なアルゴリズムを以下に示す。

【0105】アルゴリズム2

1）$n = (0, 0)$, $F = \{1, y, \cdots, y^{a-1}\}$, $G = \phi$, $A = \{x-1\}$, $B = \{u\}$

2）Bの全ての多項式の最高次係数が0でなければ、新しい△と定義点tを定める。

【0106】3）そのあらゆる定義点tにおいて以下の手続きを実行する。

【0107】①$t = (s1^{(i)}, s2^{(i)}) \rightarrow$型＜i，n2 － i＞のh4を作る。

【0108】②$t = (n1 - s1^{(i)} + 1, n2 - s2^{(i)}) \rightarrow$型＜n2 － i，i＞のh4を作る。

【0109】③$t = (n1 + 1, n2 - s2^{(i)}) \rightarrow h4 = x^{s1(n2-i)-n1-1} \cdot f^{(n2-i)}$ Bから最高次係数が0でない全ての多項式を除去し、新しく求めたh4を全てBに挿入する。

【0110】4）3）で決定された型の多項式h3を作る。また、FからBの除去した多項式に対応する多項式を除去し、新しく求めたh3をFに挿入する。

【0111】5）△が変化した場合、新しいFの補助多項式系Gの多項式を旧のGと除去したFの多項式の中から選び出す。また、新しいBの補助多項式系Aの多項式を旧のAと除去したBの多項式の中から選び出す。

【0112】6）n＝n＋1；n＝pならば終了；さもなければ2）へ

ただし、

h4　：型＜i，j＞のh4を次のように定義する。

【0113】$h4 = x^{r1-s1(i)} \cdot b^{(i)} - (d_i/d_j) \cdot x^{r1-n1+m1-t1(j)} \cdot a^{(j)}$

アルゴリズム2と神谷アルゴリズムの違いも、アルゴリズム1と阪田アルゴリズムの違いと同様に$df_n{}^{(i)}$を直接計算するか、Bに属す多項式b(i)の最高次係数とするかである。従って、アルゴリズム1と阪田アルゴリズムの間で成り立った関係が、アルゴリズム2と神谷アルゴリズムの間で成立すればアルゴリズム2によって得られるFと神谷アルゴリズムによって得られるFが同じ多項式系であることが証明できる。よって、次のようになる。

【0114】定理1の$f^{(i)} \in F$を神谷アルゴリズムの$f^{(i)}$とすれば、それに対応する多項式$b^{(i)} = f^{(i)} \cdot u$の$z^{v-n+s}$の係数は神谷アルゴリズムの$df_n{}^{(i)}$となっていることがいえる。よって、定理1はこの実施例においても成立する。

【0115】定理2は一般的に成立する。

【0116】定理3の$f_q{}^{(k)}$は神谷アルゴリズムの$f^{(i)}$であるので、定理3も同様に成立する。

【0117】従って、阪田アルゴリズムの場合と同様に$w_q{}^{(k)} = b_q{}^{(k)}$であることがいえるので、$b_q{}^{(k)}$はh4によって更新できる。この場合も、定理1から$d_i$，$d_j$は各々多項式$b_n{}^{(i)}$，$b_m{}^{(k)}$の最高次数の係数であ

る。

【0118】以上から、アルゴリズム2によって得られるFと神谷アルゴリズムによって得られるFが同じ多項式系であることが証明された。

【0119】よって、アルゴリズム2も図1のような装置によって実現できる。ただし、制御及び処理はアルゴリズム1と異なる。（処理はh3，h4の演算を行い、制御は型の判定が簡単化されている。また、メモリ部もA，B，F，Gの初期値も1）に示された多項式が設定される。）また、アルゴリズム2も並列処理に適しているので、処理回路は1つである必要はない。以上から、アルゴリズム2を実行する回路も簡単に実現することができることがわかる。

【0120】［実施例3］多次元配列を生成する記憶素子数最小の多次元シフトレジスタを合成する多項式系Fを導出するアルゴリズムもまた、阪田によって示されている[13]。このアルゴリズムは多次元BM法と呼ばれており、ここでは、多次元BM法に対応する新しいアルゴリズムを考える。

【0121】詳細なアルゴリズムは煩雑であるので、BM法と呼ばれるアルゴリズムの特徴を図で示す。1次元のBM法を含め、その拡張である阪田によって提案されたアルゴリズム（多次元BM法）は図2のような構成を持っている。それに対応するアルゴリズムは図3のようになる。図2と図3の違いも前述の2つの実施例と同様に、$df_n^{(i)}$を直接計算するか、Bに属す多項式$b^{(i)}$の最高次係数とするかである。従って、定理1～3が多次元配列においても成立すれば、図2の多次元BM法によって得られるFと図3のアルゴリズムによって得られるFが同じ多項式系であることが証明できる。

【0122】前述の実施例において点はn＝（n1，n2）となる2次元空間上で定義され、配列及び多項式の各係数はその点における写像である。これをN次元空間に拡張した場合、点はn＝（n1，n2，…，nN）と定義され、配列及び多項式もその点における写像になり、その関係は次元を問わず同じである。従って、N次元空間においても定理1～3は同様に成り立つことは明らかである。

【0123】従って、図3のアルゴリズムから生成されるFは図2のアルゴリズムから生成されるFと同じであることがいえる。よって、図3のアルゴリズムも図1のような装置によって簡単に実現できる。

【0124】ただし、図3のアルゴリズムは1次元の場合、前述のRS符号やBCH符号の復号において用いられるEu法と呼ばれるアルゴリズムになる。また、効果の所で述べるように図3のアルゴリズムは1次元Eu法の特徴を失うことなく、多次元へとEu法を拡張している。従って、アルゴリズム1，2を含む図3の構成を持つアルゴリズムを多次元BM法に対応して多次元Eu法と呼ぶことにする。

$$b_{n+1}^{(k)} = z^{rs} \cdot b_n^{(i)} - (d_i / d_j) \cdot z^{rt} \cdot a_n^{(j)} \quad (7)$$

$$b_{n+1}^{(k)} = d_j \cdot z^{rs} \cdot b_n^{(i)} - d_i \cdot z^{rt} \cdot a_n^{(j)} \quad (8)$$

以上からわかるように、Eu法とBM法の違いを次のように表せる。

【0132】BM法：$df_n^{(i)}$を直接計算して、その値を用いて$f_{n+1}^{(k)}$を求める。

【0133】Eu法：$df_n^{(i)}$を直接計算せず、多項式$b_n^{(i)}$と$b_n^{(j)}$の最高次の係数d_i，d_jを用いて式（7）のように$b_{n+1}^{(k)}$を求める。また、同じd_i，d_jを用いて$f_{n+1}^{(k)}$を計算する。ただし、$a_n^{(j)} = b_m^{(j)}$。

【0125】［その他の実施例］前述したように1次元のBM法や1次元のEu法はRS符号やBCH符号においてよく用いられているので、種々の変形アルゴリズムが提案されている。よく知られた1次元Eu法の変形アルゴリズムとして連分数法(L. R. Welch and R. A. Scholtz: "Continued fractions and Berlekamp's algorithm," IEEE Trans. Inf. Theory, IT-25, pp. 19-27, Jan. 1979)がある。これは、式（6）の演算を最後まで行わず、最終結果に関係のない演算を省いた手法である。しかし、式（6）の形式の演算によってBを更新し$df_n^{(i)}$を直接計算しないという点では同じであるので、本方式は連分数法を含む種々のアルゴリズムの変形に対しても有効であることは明かである。

【0126】また、本方式はアルゴリズム1，2の例からもわかるように型の分類の方法に依存しない。従って、本方式は多次元配列に対してBM法のように直接$df_n^{(i)}$を計算せず、式（6）の形式の演算によってBを更新し、$df_n^{(i)}$をBの多項式の最高次係数として演算する手法に対して全て有効である。

【0127】また、点や各パラメータ（rs，rt等）は整数とし、$f_n^{(i)}$及び$b_n^{(i)}$は多項式としているが、点や各パラメータを任意の複素数、多項式を有理関数等に拡張しても本方式が有効であることは明かである。

【0128】また、多項式uやfの次数を逆（双対多項式）にすれば、多項式b＝f・uも次数が逆になりd_iを多項式bの最低次係数とすることもできる。

【0129】また、定理3においてd＝$z^{v} \cdot x$としているが、d＝z^{v+1}またはd＝0としても$w_q^{(k)} = b_q^{(k)}$となり図3の構造を持つアルゴリズムの結果は変わらないことは明かである。

【0130】また、以上の実施例ではBの更新のために次の式（7）のような演算を用いているが、式（8）のような演算によっても定数項の違いを除いて同じ結果が得られる。ただし、$a_n^{(j)} = b_m^{(j)}$。

【0131】

【0134】

15　　16

$$b_{n+1}^{(k)} = z^{rs} \cdot b_n^{(i)} - (d_i / d_j) \cdot z^{rt} \cdot a_n^{(j)} \quad (7)$$

この違いから次のような相違点がＢＭ法とＥｕ法の装置化及び高速化に対して生じる。

【０１３５】ＢＭ法：　ＢＭ法は$f_n^{(i)}$と$f_{n+1}^{(k)}$を並列に計算することは困難である。なぜならば、$f_{n+1}^{(k)}$を計算するためには$df_n^{(i)} = d_i$が必要であり、$df_n^{(i)}$は$f_n^{(i)}$の全ての係数を用いて計算される。従って、$df_n^{(i)}$が得られるのは$f_n^{(i)}$の計算が終わった後であるので、$f_{n+1}^{(k)}$の計算は$f_n^{(i)}$の計算が終わった後でしか始まらない。従って、図４(a)に示すようにＢＭ法では点ｎとｎ＋１における多項式$f_n^{(i)}$と$f_{n+1}^{(k)}$は逐次的に計算される。さらに、ＢＭ法は$f_n^{(i)}$と$f_{n+1}^{(k)}$及び$df_n^{(i)}$の演算を１クロックで計算したとしても、それらが並列に実行されないため図４(b)に示すように２ｐクロック程度の計算時間が必要である。これは、１次元ＢＭ法から図２に示される多次元ＢＭ法に対して共通する特徴である。

【０１３６】Ｅｕ法：　Ｅｕ法は$f_n^{(i)}$と$f_{n+1}^{(k)}$を並列に計算することができる。なぜならば、Ｅｕ法においてはd_iは$b_n^{(i)} \in B$の最高次係数となっており、Ｂに属す多項式は式（７）によってＢに属す多項式のみによって計算される。$b_n^{(i)}$と$a_n^{(j)}$を用いて式（７）が最高次数から下位次数の方へ順次計算されているとすると、出力$b_{n+1}^{(k)}$は最高次係数から下位次係数の方へ順次得られる。よって、点ｎ＋２における多項式$b_{n+2}^{(k)}$は$b_{n+1}^{(i)}$の最高次係数が得られた時点から計算を始めることができる（$a_{n+1}^{(j)}$は既に得られている多項式である。）。これは点ｎ＋１とｎ＋２における多項式が並列に計算できることを示しているが、点ｎとｎ＋１においても同様であることは明かである。従って、Ｂの更新のための演算は各点において並列に実行することができる。また、$f_{n+2}^{(k)}$と$f_{n+1}^{(k)}$の計算も$b_{n+1}^{(i)}$と$b_n^{(i)}$の最高次係数がわかった時点で始めることができるので、図５の(a)に示すように$f_{n+1}^{(k)}$の演算も各点において並列に実行できる。よって、Ｅｕ法は式（７）の演算を行う処理回路を複数用意すれば、その数に比例した高速化が容易に行える。これは、よく知られた１次元Ｅｕ法から図３の多次元Ｅｕ法に対して共通の特徴であり、上記のＢＭ法にはない特徴である。また、$b_n^{(i)}$，$b_{n+1}^{(k)}$及び$f_n^{(i)}$，$f_{n+1}^{(k)}$の演算を１クロックで計算した場合、図５の(b)に示すようにｐクロック程度の計算時間で良く、ＢＭ法より高速な演算が行える（ただし、図５は点ｎにおけるd_iをd_nと表す。）。

【０１３７】一般的に、高速なアルゴリズムとは計算量の少ないアルゴリズムを指す場合が多い。しかし、高速化を実現する場合、アルゴリズムの計算量を減少させるアプローチとアルゴリズムの並列度を高めるアプローチが考えられる。それは並列処理によって複数の処理を同時に実行すれば処理時間が短縮できるためである。最近はＶＬＳＩ技術の進歩により大規模な並列処理チップを容易に実現することができる。よって、並列度の高いアルゴリズムの方が高速処理向きである場合も多い。

【０１３８】

【発明の効果】以上説明したように、本発明によるＥｕ法は、ＢＭ法よりも並列度が高く、高速処理に向いていると言える。また、ＢＭ法が多項式$f_n^{(i)}$とスカラ量$df_n^{(i)}$という２種類の演算を必要とするのに対して、Ｅｕ法は$f_n^{(i)}$，$b_n^{(i)}$に対して式（７）に示される同一形式の演算のみでよいので、処理回路をユニット化しやすくより並列処理に向いていると言える。

【０１３９】従って、並列処理を用いて装置化する場合、本発明によれば、簡単に高速な処理装置を構成できるという効果がある。

【図面の簡単な説明】

【図１】本発明による多次元配列生成回路である。

【図２】従来のＢＭ法による多次元配列生成アルゴリズムである。

【図３】本発明によるＥｕ法による多次元配列生成アルゴリズムである。

【図４】従来のＢＭ法の動作説明図である。

【図５】本発明によるＥｕ法の動作説明図である。

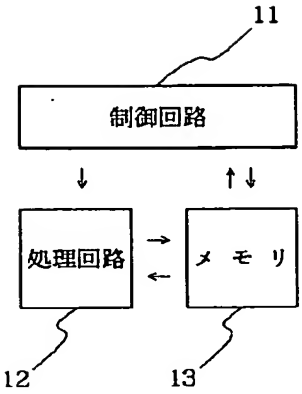
【符号の説明】

１１　制御回路

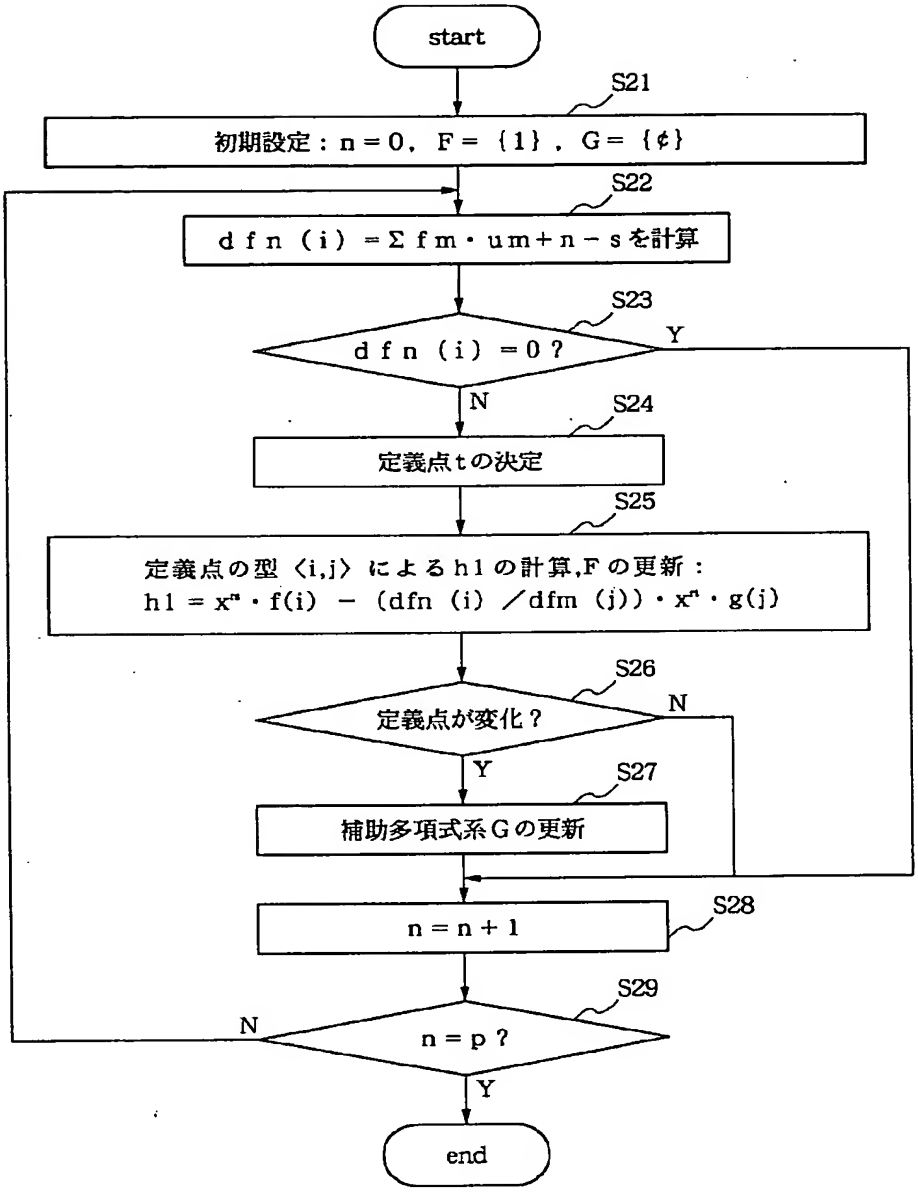
１２　処理回路

１３　メモリ

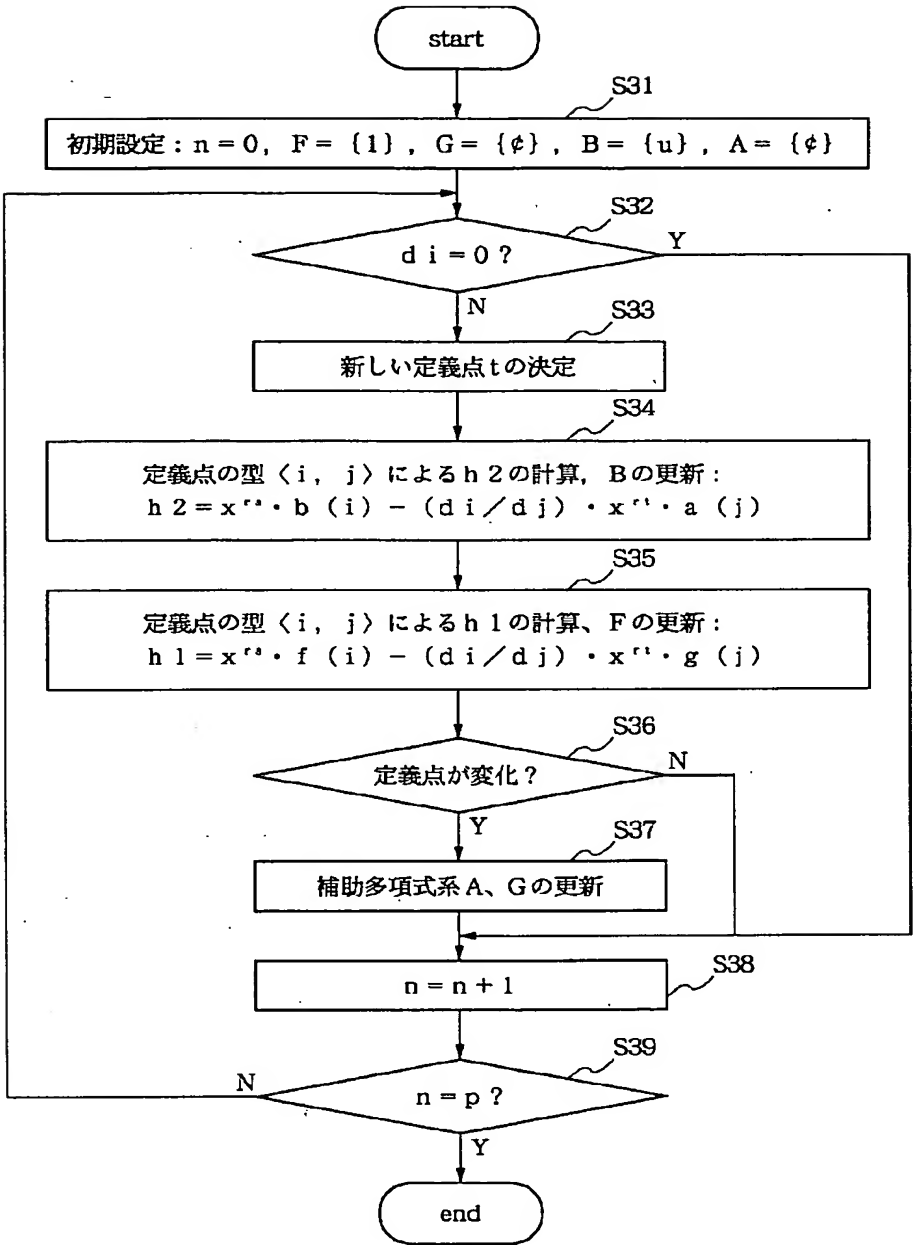
【図１】


11
制御回路
処理回路
メ モ リ
12
13

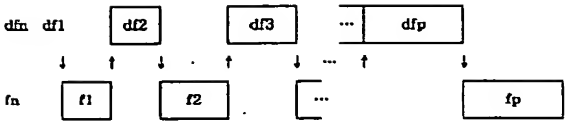
【図２】


start
S21
初期設定：n＝0，F＝｛1｝，G＝｛¢｝
S22
d f n（i）＝Σ f m・u m＋n－s を計算
S23
d f n（i）＝0？
Y
N
S24
定義点 t の決定
S25
定義点の型〈i,j〉による h1 の計算,F の更新：
h1 = x^n・f(i) －（dfn（i）／dfm（j））・x^n・g(j)
S26
定義点が変化？
N
Y
S27
補助多項式系 G の更新
n＝n＋1
S28
S29
n＝p？
N
Y
end

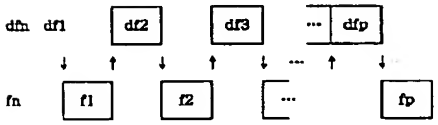
【図３】


start
S31
初期設定：n＝0，F＝｛1｝，G＝｛¢｝，B＝｛u｝，A＝｛¢｝
S32
d i＝0？
Y
N
S33
新しい定義点tの決定
S34
定義点の型〈i，j〉によるｈ２の計算，Bの更新：
h 2＝x^δ・b（i）－（d i／d j）・x^τ・a（j）
S35
定義点の型〈i，j〉によるｈ１の計算、Fの更新：
h 1＝x^δ・f（i）－（d i／d j）・x^τ・g（j）
S36
定義点が変化？
N
Y
S37
補助多項式系A、Gの更新
n＝n＋1
S38
S39
n＝p？
N
Y
end

【図４】

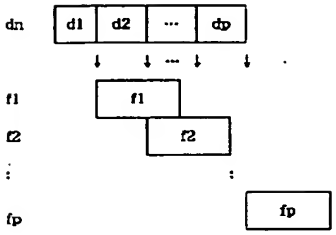

dfn df1 df2 df3 ··· dfp
fn f1 f2 ··· fp

(a)

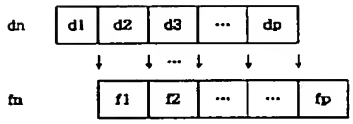

dfn df1 df2 df3 ··· dfp
fn f1 f2 ··· fp

(b)

【図５】


dn d1 d2 ··· dp
f1 f1
f2 f2
: :
fp fp

(a)


dn d1 d2 d3 ··· dp
fn f1 f2 ··· ··· fp

(b)